PROCEEDINGS OF

THE IEEE INTERNATIONAL CONFERENCE
ON

RESEARCH CHALLENGES IN INFORMATION SCIENCE

RCIS 2009

Fez, April 22-24, Morocco

EDITORS:
André Flory
Martine Collard

SPONSORSHIP

We wish to thank the following organizations for their contribution to the success of this Conference.

INTRODUCTION

The series of international conferences on Research Challenges in Information Science (RCIS) aims at providing an international forum for scientists, researchers, engineers and developers from a wide range of information science areas to exchange ideas and approaches in this evolving field. While presenting research findings and state-of-art solutions, authors are especially invited to share experiences on new research challenges.

It is a big pleasure for all of us to celebrate this third edition of RCIS for which we sought research focusing on foundational or technological aspects as well as research based on experience and describing industrial aspects on the following main topics: Databases, Information Systems, Web Systems, Business Process Modelling, Analysis and Design, Intelligent Agents, Ontologies and Knowledge Discovery from Data.

Of course, we express our appreciation to all Program Committee members for their involvement in the evaluation of submitted papers. It is not an obvious task to select a relevant subset of submitted works, but with their support, we have been able to prepare an exciting scientific program.

For this third edition, we received 112 submissions. All submitted papers were carefully evaluated based on originality, significance, technical soundness, and clarity of expression by three reviewers. From these submissions, 35 were selected as long papers and 15 as short papers to be presented at the conference. Additionally, 9 doctoral papers were accepted.

In addition, the conference program is properly complemented with three renowned keynote speakers, **John Mylopoulos** (University of Toronto, USA), **Oscar Pastor** (University of Valencia, Spain) and **Aris Ouksel** (University of Chicago, USA). Thanks to all of them for sharing this event with us. This is for sure a strong added value for this third edition of RCIS.

Thank you very much to local organizers for making easier to solve any problem. RCIS 2009 would not have been possible without their big effort ; they selflessly offered their time and energy to make this conference a success.

Finally we welcome you to Fez, we hope you enjoy this conference and benefit from the presentations which are made.

André Flory
Martine Collard
Fez, Morocco, April 2009

Table of Contents

DOCTORAL PAPERS

A Recommendation Algorithm for Knowledge Objects based on a Trust Model

Aurora Vizcaíno, Javier Portillo-Rodríguez, Juan Pablo Soto, Mario Piattini
Alarcos Research Group
University of Castilla – La Mancha
Ciudad Real, Spain
{aurora.vizcaino, javier.portillo}@uclm.es
juanpablo.soto@inf-cr.uclm.es
mario.piattini@uclm.es

Oliver Kusche
Institute for Applied Computer Science (IAI)
Forschungszentrum Karlsruhe
Karlsruhe, Germany
oliver.kusche@iai.fzk.de

Abstract—**This paper presents a recommendation algorithm with which to recommend Knowledge Objects within a Community of Practice (CoP). It is based on a trust model that takes into account not only previous experience of a Knowledge Source, but also the position of the respective member in the community, along with that member's level of expertise. Furthermore, the system attempts to emulate the principle of human intuition. The combination of these factors, which can be adjusted by means of weight factors, enables the system to make decisions based on a calculated trust value even if a new member is introduced into the community or the entire community is newly created. To demonstrate the capabilities of this algorithm, a tool employing a multi-agent architecture has been developed which is also presented.**

Keywords: Knowledge Management, Communities of Practice, Recommender Systems, Trust Models

I. INTRODUCTION

Knowledge Management (KM) is an important success factor for any company. The purpose of knowledge management is to help companies to create, share and use knowledge more effectively [6]. Information technologies play a key role in achieving these goals. However, it is rather difficult to design tools which, for instance, recommend knowledge since it is first necessary to know certain characteristics of the person or entity that will receive the recommendation. The domain in which the knowledge will be used must also be taken into account. Therefore, before designing a knowledge recommendation tool, we first studied how people share knowledge in order to discover what factors influence this process. This study led us to the realisation that employees frequently exchange knowledge with people who work on similar topics and that communities are consequently either formally or informally created. These can be called "communities of practice", by which we mean groups of people with a common interest in which each member contributes knowledge concerning a common domain [12].

Communities of practice (CoPs) enable their members to benefit from each other's knowledge. This knowledge resides not only in people's minds but also in the interaction between people and documents. CoPs share values, beliefs, languages, and ways of doing things. Many companies report that such communities help to reduce problems caused by a lack of communication, and save time by "working smarter" [13].

We are therefore developing a tool to support knowledge sharing in CoPs. The first problem that we encountered when designing this tool was how to evaluate how useful a piece of knowledge is. Furthermore, we identified the need to encourage the reuse of information. In order to confront these two issues an algorithm to recommend Knowledge Objects (KO) was developed. This uses a trust model that we had previously developed to discover how trustworthy a Knowledge Source (KS) is.

The remainder of this work is organized as follows: The next section outlines how the recommender system works; this information is necessary if we are to understand the algorithm used to recommend KO. Said algorithm is then described in Section 3. In Section 4 we describe related work. Finally, in Section 5, conclusions and future work are outlined.

II. A RECOMMENDER SYSTEM

This section describes the recommender tool. This tool uses a multi-agent architecture (see [11]) in which each CoP member is represented by a software agent called a User Agent. A new community member must first join a community, and this is done by using the "*Register*" menu and choosing a community from those which are available. Once registered, a member can provide new KOs or use KOs which are already available in the community and propose new topics. The two first situations are described.

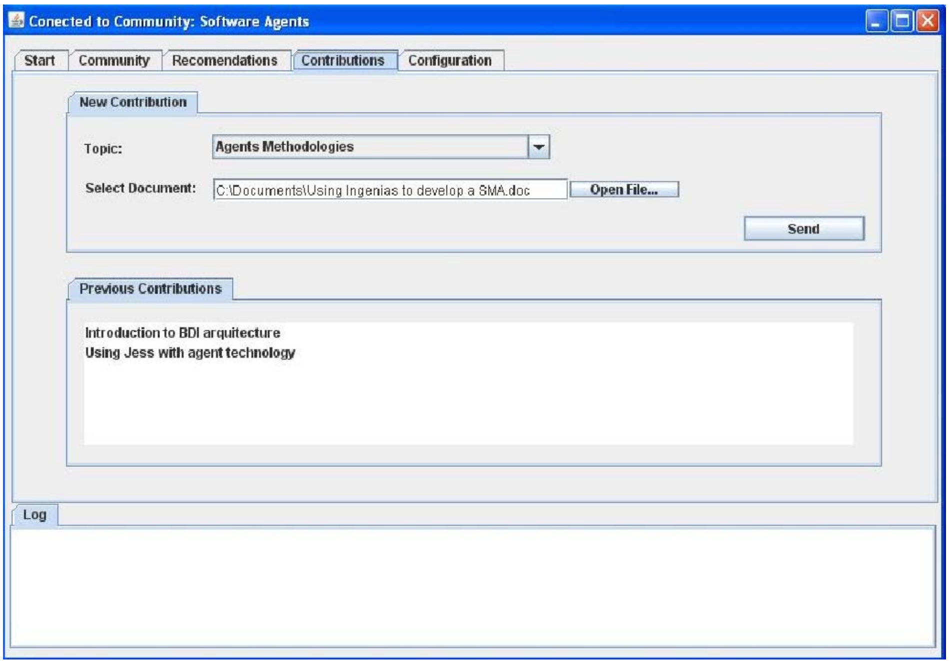

Figure 1. Interface of the tool

1) Proposing a new KO. In order to provide a KO (for instance a document) a person must use the "Propose" menu and must configure the followings options (see Figure 1):

- Topic: In each community there may be different topics or areas. The users can choose that in which they intend to propose the document.
- KO: The proposed document.

Once the user has chosen the options, the User Agent sends the values to another software agent called the Manager Agent which is in charge of adding the new KO to the community and modifying the frequency of contribution of the User Agent in this community.

2) Using community KO. Members can search for a KO in every community in which they are registered and their User Agent will help them to find that which is most suitable. Therefore, when someone searches for a KO relating to a topic their User Agent consults the Manager Agent about which KOs are related to this topic. The Manager Agent then replies with a list of KOs. The User Agent sorts this list by using an algorithm which will be explained in the following section of this paper. The User Agent can therefore detect how worthy a KO is, thus saving employees time, since they do not need to review all the KOs related to a topic but only those considered to be most relevant by the members of the community or by the user him/herself.

Once one or several KOs have been chosen, the user must then evaluate the KOs consulted in order to provide feedback to the community about them.

III. Description of the Algorithm

This section describes the algorithm used for the tool to recommend a KO. The input of this algorithm will be a set of KOs. Each KO may or may not have been evaluated previously, so a KO may already have a list of evaluations (along with the identity of each person who evaluated it), or it may appear without any evaluation. This aspect will be taken into account by the algorithm which therefore distinguishes two groups:

Group 1 (G1): This group is formed of the KOs that have been evaluated. This is the most important group since if we have previous evaluations about a KO we have more information about it in order to know whether it is advisable to recommend it or not.

Group 2 (G2): these KOs have not been used previously so the tool does not have any evaluations about them. Let us now observe how each group is processed by the algorithm.

In G1 the KOs will be ordered by a Recommendation Rate which is calculated for each KO. Hence RR_k signifies the Recommendation Rate for a particular KO called k, and is obtained from:

$$RR_k = w1 * TE_i + w2 * TS_{ik} \tag{1}$$

where TE_i is the mean of the evaluations determined by the trust that an agent "i" has in each evaluator (the person who has previously evaluated that KO). TE_i is calculated as:

$$TE_i = \frac{\sum_{j=1}^{n} E_{jk} * TS_{ij}}{\sum_{j=1}^{n} TS_{ij}} \tag{2}$$

Therefore, TS_{ij} is the trust value that the User Agent "*i*" has in the knowledge source "*j*", since in a CoP the source which provides a KO will usually be a CoP member. TS_{ij} represents the trust that an agent "*i*" has in another agent "*j*" and E_{jk} is the evaluation that an agent "*j*" has made about a particular KO "*k*".

The parameter TS_{ik} used in Formula (1) similarly indicates the trust that an agent "*i*" has in a knowledge source "*k*". Both w1 and w2 are weights which are used to adjust the formula, and will be discussed in more detail later. The sum of w1 and w2 should be 1.

In order to illustrate how all the parameters are obtained, consider the following example, in which the tool has the evaluations of a KO called *k,* as is shown in Table (1).

TABLE I. EVALUATIONS OF A KO (X)

Evaluator	Trust Value (TS_{ij})
User 1	5
User 23	1
User 4	2
User 13	1

The first step is to calculate TE_i. When we first began to design the algorithm, TE_i was calculated as the average of these four values. However, after testing the recommender system we realized that the average did not give good results since the situation sometimes arose that, for example, User 1 was an expert in the topic of a particular KO, so the expert's opinion should have had more weight than for instance that of novel members who did not have as much knowledge about that topic. We therefore decided to use a weighted mean in which each evaluation is carried out according to the trust value that the User Agent has of each evaluator (more details of how this value is obtained will be given later). Table 2 shows an example of the trust values that a particular User Agent may have with regard to the User Agents that have evaluated the KO.

TABLE II. EVALUATOR'S TRUST VALUES

Evaluator	Trust value (TS_{ij})
User 1	5
User 23	1
User 4	2
User 13	1

The result of applying the weighted mean in this case is:

$$TE_i = \frac{5*5+2*1+1*2+1*1}{5+1+2+1}$$

$$= \frac{25+2+2+1}{9}$$

$$= 3.33$$

as opposed to the average which would be 2.25 (from (5+1+2+1)/4). This shows that the result is more trustworthy with the weighted mean as it is closer to the expert's opinion whose evaluation was 5. The second step is to discover the TS_{ik} in Formula (1) in order to know how trustworthy the source of this KO "k" is. Let us suppose that TS_{ik} is equal to 5 (an explanation of how the *TS* values are obtained will be provided in the second part of this section). It is now therefore necessary to consider what values w1 and w2 should have. One advantage of this formula is that it permits us to change these weights in accordance with the CoPs' preferences, since some CoPs may prefer not to take the TS_{ik} into consideration, and in this case w2 would be zero. Other CoPs might wish to give a little weight to this factor and more weight to $TE_{i,}$ so w1 could be 0.8 and w2 0.2. These weights therefore give more importance (more weight) to the trust obtained by taking into account previous evaluations. In this case the results obtained are:

$$RR_k = 0.8*3.33+0.2*5 = 3.66$$

So the Recommendation Rate for this KO "k" would be 3.66.

The algorithm would then calculate the RR of each KO related to a topic that a user is interested in and would later show a list with the KOs ordered according to the RR. In the case of there being a high quantity of KOs, then only those with a higher RR would be shown.

Group 2 will use another formula to calculate the RR for each KO since in this case there are no results of previous evaluations of the KOs. The formula used is, therefore:

$$RR_k = w1 * TS_{ix} + w2 * Re_x \tag{3}$$

where TS_{ix} is the Trust that the User Agent "i" has in the KS "x" which provides the KO "k", and Re_x is the reputation that the KS has (according to another member's agents' opinion). This Re_x value is calculated by asking those agents with a higher trust value about the KS with a weighted mean which is subsequently calculated. Re_x is therefore obtained as:

$$Re_x = \frac{\sum_{j=1}^{n} TS_{jx} * TS_{ij}}{\sum_{j=1}^{n} TS_{ij}} \tag{4}$$

where TS_{jx} is the trust that an agent "*j*" has in the KS "x" and TS_{ij} is the trust value that the agent "*i*" has in agent "*j*". Therefore, the agent's opinion of KS "x" is adjusted by the opinion that the agent "*i*" has with regard to the agent which is giving its "opinion" (trust value in the KS "x").

In this example, the values 0.6 and 0.4 have been chosen for the weights w1 and w2 respectively, thus placing more importance on the trust value that an agent itself has in the KS than on the other member's opinion. However, this is also useful to adjust the RR_k.

In order to illustrate how RR_k is calculated, let us imagine that we wish to discover the RR_k of a KO provided by Agent 23. Table 3 shows the trust value that a User Agent, for instance Agent 7, might have about other agents.

TABLE III. AGENT 7'S TRUST VALUES FOR OTHER AGENTS

Agents	Trust ($TS_{i=7,j}$)
Agent 14	4.3
Agent 23	2.4
Agent 2	2.3
Agent 32	1.9
Agent 12	1.8
Agent 6	2.1
Agent 8	2.8

We can see that the value of $TS_{7,\,23}$ for Agent 23 is 2.4. The User Agent now asks the four Agents with the highest trust values for their opinion of Agent 23 (trust values). Let us suppose that these values are those shown in Table 4.

TABLE IV. AGENTS WITH THE HIGHEST TRUST VALUES FOR AGENT23

Agent	Trust in Agent 23 ($TS_{j,x=23}$)
14	1
2	3.4
6	1.2
8	1.3

Therefore, Re will be calculated as:

$$Re = \frac{1*4.3+3.4*2.3+1.2*2.1+1.3*2.8}{4.3+2.3+2.1+2.8}$$

$$= \frac{4.3+7.82+2.52+3.64}{11.5}$$

$$=1.58$$

We can thus obtain the *RR* for this particular KO, as:

$$RR = 0.6*2.4+0.4*1.58$$

$$= 1.44+0.63$$

$$= 2.07$$

Note that the obtained value is smaller than if we had only considered the $TS_{7,\,23}$ (2.4). The User Agent will then show another list with the KOs that a higher *RR* has obtained (see Figure 2).

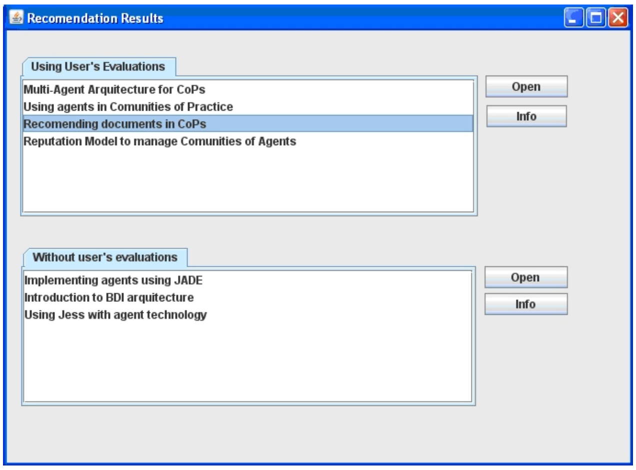

Figure 2. List of documents recommended

The algorithm can be summarized in the following pseudocode:

```
Begin
while KO_list is Not empty
      For each KO
          if(KO.hasEvaluations()) then
               KO.Calculate_RR(evaluations)
               G1.Add (KO)
               KO_List.Remove (KO)
          else
               KO.Calculate_RR()
               G2.Add (KO)
               KO_List.Remove(KO)
      End_for
End_while
G1.Order()
G2.Order()
End
```

In this pseudocode, each KO in the list is tested to discover whether it has evaluations in order to know whether it belongs to Group 1 (G1) or Group 2 (G2). In each case the *RR* is calculated by using the formulas explained previously and the KO is removed from the original list. Finally G1 and G2 are order by the *RR* and the KOs with the highest *RR* will be recommended.

The second part of this section explains how a User Agent calculates TS_{ij}, for which the following formula is used:

$$TS_{ij} = wp*P_j + we*LE_j + wi*I_{ij} + PE_{ij} \qquad (5)$$

Each factor of the formula is presented and the description of each one is explained as follows: P_j is the Position of the Agent "*j*" in the CoP or in the organization in which the CoP exists. LE_j is the Level of Expertise that the person represented by the Agent "*j*" has in a particular domain. I_{ij} is the intuition that the User Agent "*i*" has with regard to the Agent "*j*" and finally PE_{ij} is the value of Previous Experience that the Agent "*i*" has had with the Agent "*j*".

Position: When a new member joins a community that person must indicate their position within the organization and their User Agent will calculate the Position (P) value of that person by using the following formula:

$$P = \frac{\mathrm{UPL}}{NL} \tag{6}$$

UPL = User's Position level
NL = number of levels in the community

Therefore, if a community has, for example, 5 possible position levels (NL=5), and if the new member has a level of UPL=2 then the value of P will be 2/5=0.4. Therefore, the different values of P for a community with five levels will be those shown in Table 5:

TABLE V. EXAMPLE OF POSITION VALUES

Levels	Values P
1	0.2
2	0.4
3	0.6
4	0.8
5	1

The P values will always be between 0 and 1. Moreover, situations may exist in which P will not be taken into account, for instance in those CoPs in which all the members have the same level or whose members do not wish to consider this criterion. In these cases wp (weight of position) will be zero and position will not be considered in Formula 5. Another situation exists in which wp is equal to zero. This occurs when the value of the Previous Experience PE > U (U being a threshold which is chosen when creating the community). In this case, the User Agent will use the following formula to calculate the wp value:

$$Wp = \mathrm{int}\,(U/PE_{ij}) \text{ being } PE_{ij} > 0 \tag{7}$$

U = Threshold of Previous Experience
PE_{ij} = Value of Previous Experience of an agent *i* with regard to another agent *j*.

Therefore, when PE_{ij} is greater than a particular threshold U, wp will be 0, thus ignoring the position factor. However, when one agent does not have enough Previous Experience (PE) of another it may use other factors to obtain a trust value. On the other hand, when the agent has had a considerable amount of previous experience with this agent or with the knowledge that it has provided, it is more appropriate to give more weight to this factor, since previous experience is the key factor in all trust models, as will be described in Section 4. Therefore, if an agent *j* has a high value of position but most of agent *i*'s previous experience of *j* has not been successful then the position will be ignored. This thus avoids the situation of, for instance, a superior who does not contribute with valuable documents but is considered trustworthy solely because of being a superior.

Level of Expertise (LE): This factor is used to represent the level of knowledge and know-how that a person has in a particular domain. This factor may change since a person may become more expert in a topic in the course of time.

In the tool presented, when creating a community the levels of expertise considered is also indicated, for instance: novice, beginner, competent, expert and master. Each time a new member joins a community they will indicate the level of expertise that they consider theirselves to have. If the members of the community and their level of expertise are known to the creator of the community then that person can introduce them in the tool. Once the level of expertise has been introduced, the user agent will calculate the value for this level by using the following formula:

$$LE = L/NT + AV_j \tag{8}$$

where L is the level of expertise that was introduced, and NT is the number of levels in the community. The term AV_j is the Adjustment Value for agent j. This term is particularly important since it will be used to adjust the experience of each user. This was introduced with the goal of avoiding two situations:

- A person either intentionally or mistakenly introduces a level of experience that is not his/her true level.
- Whilst in the community, persons become more expert, leading to the situation that their level of expertise should be adjusted.

AV_j will initially be 0, and each time a member interacts with a KO provided by j the member will rate this KO and send this evaluation to the manager agent in charge of managing the community. The Manager Agent will verify whether the evaluation is negative or positive. If it is positive, then agent j's level of experience can be modified by calculating AVj as:

$$AV_j = (VL_n - VL_{n-1})/PT \quad (n \neq 1) \tag{9}$$

If it is negative, then:

$$AV_j = -(VL_n - VL_{n-1})/PT \quad (n \neq 1) \qquad (10)$$

where VL_n is the value that a particular level of experience has. PT is the Promotion Threshold which is used to determine the number of positive rates necessary to promote a superior level of experience. Let us illustrate this with an example. In a community there are four levels with the following values:

TABLE VI. POSITION LABELS

Labels	Level(n)	Value(VL)
Beginner	1	0.25
Competent	2	0.50
Expert	3	0.75
Master	4	1.00

In this case, the difference between the levels is 0.25 as:

$$VL_n - VL_{n-1}=0.25$$

In this version of the tool it is assumed that at least 5 ratings are necessary to change the level, so PT will be 5 and AV_j will be 0.25/5=0.05. This is therefore the value that will be added when a positive rating is received or that will be subtracted when this rating is negative. With five positive ratings (5*0.05=0.25) there is thus a level promotion.

Intuition: This term is used when the Previous Experience is low and it is necessary to use other factors to calculate a trust value. This is one contribution of our work, since most of the earlier trust models are based solely on previous experience. The agents attempt to emulate human behaviour, as people often trust more in people who are similar to themselves. For instance a person who has to choose between information from two different people will normally choose that which comes from the person who has the same background, same customs etc. By following this pattern, the agents compare their own profiles with the other agents' profiles in order to decide whether a person appears to be trustworthy or not. Therefore, the more similar the profiles of two agents are, for instance *i* and *j*, the greater the I_{ij} value in Formula (5) will be. Consequently, when an agent determines that it does not have sufficient data in order to decide whether or not to trust another agent, it may still decide that the other agent seems trustworthy because of its similar properties. The agents' profiles may differ according to the community in which they are working. In our case, as the data stored in the agents' profiles are 'position' and 'expertise', both these features will be taken into account. Therefore, the factors that the tool compares are:

- Experience Difference (ED)
- Position Difference (PD)

Thus, the Intuition value of an agent *i* with regard to *j* (I_{ij}) is:

$$I_{ij} = ED_{ij} + PD_{ij} \qquad (11)$$

where $ED_{ij} = LE_i - LE_j$ and $PD_{ij} = P_i - P_j$

This formula is based on the idea that a person normally has a greater level of trust in people who have a higher level of experience or who are in a higher position than that person. Hence, when an agent compares its profile with another agent with higher values, the value of intuition will be positive. Let us consider the case of agent *i* which has values of LE_i=0.2 and P_i=0.6. This agent wishes to know how trustworthy another agent *j* is. In this case the agent will use Formula (5) and, depending on the information that it has about *j*, it will or will not be necessary for it to calculate the intuition factor. In this situation we shall suppose that there is little previous experience and that this must be calculated. The values for the agent *j* are LE_j=0.5 and P_j=0.5. As Figure 3 shows:

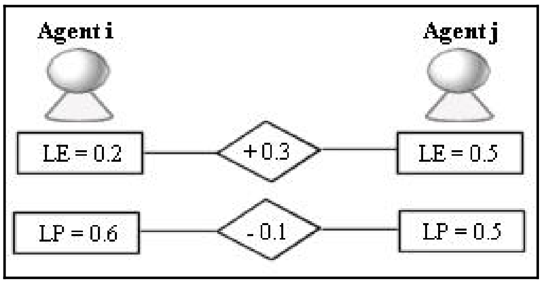

Figure 3. Comparing profiles

I_{ij}=0.2 (obtained by using formula 11) as ED_{ij}=0.3 and PD_{ij}=-0.1.

As with position, intuition will or will not be calculated depending on the level of PE (previous experience). Thus, the weight of intuition, (see Formula 5) wi will be calculated as follows:

$$wi = \text{int}(U/PE_{ij}) \text{ with } PE_{ij} \neq 0 \qquad (12)$$

Previous Experience: This factor is the most decisive of all the factors in Formula (5). In fact, all the previous factors depend on it as an agent will decide whether or not to use the remaining factors according to the value of Previous Experience (PE). Previous Experience is obtained through the interactions that the agent itself has, so this is direct experience. Each time one agent interacts with another (by interacting we mean, for instance, that one agent uses a KO provided by another), the first agent asks its user to rate that KO in order to discover whether the KO was:

- useful
- related to the topic at hand
- recommendable for other people interested in the same topic
- up-to-date

The agent then labels this interaction with a label from Table 7. A value for Current Experience (CE) is thus obtained which will modify the previous value of PE in accordance with the following formula:

$$PE_{ij}(x) = PE_{ij}(x-1) + CE_{ij}(x) \quad (13)$$

TABLE VII. PE LABELS

Label	PE Level
Very Bad	- 0.3
Bad	- 0.2
Medium	+ 0.1
good	+ 0.2
Very good	+ 0.3

where $PE_{ij}(x)$ is the value of Previous Experience that the agent i has about another agent j in an interaction x.

$EP_{ij}(x-1)$ is the value of Previous Experience that the agent i had about another agent j before the interaction x.

$CE_{ij}(x)$ is the value of the experience that i has had with j in the interaction x.

For instance, if an agent i has just taken part in an interaction with the agent j, and this is labeled as "bad", but the value of $PE_{ij}(x-1)$ was 0.8, then the value of $PE_{ij}(x)$ will be 0.6 obtained from (0.8+(-0.2)). Moreover the agent i will send the Manager Agent the value of $CE_{ij}(x)$ in order to calculate AV_j (see Level of Expertise).

As has previously been explained, the Position and Intuition factors depend on the PE value. When an agent has sufficient PE then Position and Intuition can be ignored, and only the PE and the Level of Expertise will be considered. The latter is also included to ensure that an agent takes advantage not only of its own previous experience but also of that of the other agents since Level of Expertise (LE) is adjusted by the AV_j which comes from other previous experience.

IV. RELATED WORK

Our work can be compared with other proposals that use agents and trust models in knowledge exchange. With regard to trust, in models such as eBay [7] and Amazon [2], which were proposed to resolve specific situations in online commerce, the ratings are stored centrally and the reputation value is computed as the sum of those ratings over six months. Thus, reputation in these models is a single global value. These models are too simple in terms of their trust values and the way in which they are aggregated, but they work quite well as regards giving guidance to their clients. In [14] the authors present the Sporas model, a reputation mechanism for loosely connected online communities in which, among other features, new users start with a minimum reputation value, the reputation value of a user never falls below the reputation of a new user and users with very high reputation values experience much smaller rating changes after each update.

The problem with this approach is that when somebody has a high reputation value it is difficult to change this reputation, or the system needs a high amount of interactions. A further approach of the Sporas authors is Histos which is a more personalized system than Sporas and is oriented towards highly connected online communities. In [10] the authors present another reputation model called REGRET in which the reputation values depend on time: the most recent rates are more important than previous rates. In [5] the authors present the AFRAS model, which is based on Sporas but uses fuzzy logic. The authors present a complex computing reputation mechanism which handles reputation as a fuzzy set, while decision making is inspired in a cognitive human-like approach. In [4] the authors present a trust and reputation model that considers trust and reputation as emergent properties of direct interactions between agents, based on multiple interactions between two parties. In this model, trust is a belief an agent has about the performance of the other party to solve a given task, according to its own knowledge. In [1] the authors propose a model which allows agents to decide which agents' opinions they trust more and to propose a protocol based on recommendations. This model is based on a reputation or word-of-mouth mechanism. The main problem with this approach is that every agent must maintain rather complex data structures which represent a kind of global knowledge about the whole network.

Barber and Kim present a multi-agent belief revision algorithm based on belief networks [3]. In their model the agent is able to evaluate incoming information, to generate a consistent knowledge base, and to avoid fraudulent information from unreliable or deceptive information sources or agents. This work has a similar goal to ours. However, the means of attaining it are different. In Barber and Kim's case reputation is defined as a probability measure, since the information source is assigned a reputation value of between 0 and 1. Moreover, every time a source sends knowledge, that source should indicate the certainty factor that the source has of that knowledge. In our case, the focus is very different since it is the receiver who evaluates the relevance of a piece of knowledge rather than the provider as in Barber and Kim's proposal.

In [9] the authors present a trust and reputation model which integrates a number of information sources in order to produce a comprehensive assessment of an agent's likely performance. In this case the model uses four parameters to calculate trust values: interaction trust, role-based trust, witness reputation and certified reputation. We use certified reputation when an agent wishes to join a new community and uses a trust value obtained in other communities, but in our case this certified reputation is made up of the four previously explained factors and is not only a single factor.

Also, works such as [8] use the term 'Community' to support knowledge management but a specific trust model for communities is not used.

The main differences between these reputation models and our approach are that these models need an initial number of interactions to obtain a good reputation value and it is not possible to use them to discover whether or not a new user can be trusted. A further difference is that our approach is oriented towards collaboration between users in CoPs and to encourage the reuse of information by recommending the most suitable KO for each CoP member. Other approaches are more oriented towards competition, and most of them are tested in auctions.

V. Conclusions and Future Work

This paper describes an algorithm with which to recommend KOs in CoPs. This algorithm has several advantages:

- The User Agents recommend the KO which is most suitable for their user, considering personal factors such as the previous experience that the user has had, or intuition.
- The proposed algorithm can calculate a trust value even though the CoP has only recently been created since, in order to calculate trust, various known factors are used such as Position, Level of Expertise and even Intuition. This is a key difference with regard to other algorithms which use only previous experience and which cannot then calculate trust values if the system is just starting to work. When a new member arrives it is also impossible for other algorithms to calculate a previous trust value related to this new member.
- The fact of using intuition is another important difference with regard to other algorithms. This factor has been included since we wished to imitate how human beings decide whether or not it is appropriate to trust somebody, based on intuition.
- A further contribution of this algorithm is that it is quite flexible since in many situations weights are used to modify the formulas. The designers of other recommender systems could therefore use this algorithm and decide what values they should give to these weights in order to adapt the formula to their needs.

As future work we are currently searching for other functionalities for our recommender, such as the detection of experts in a topic, since people who contribute with the most useful KO could, at first sight, be considered experts in that topic. However, various tests must be carried out in order to verify how the tool and the algorithm might be improved.

Acknowledgment

This work is partially supported by FABRUM project, Ministerio de Ciencia e Innovación (grant PPT-430000-2008-063), the MELISA (PAC08-0142-3315) and MECENAS (PBI06-0024) projects, Junta de Comunidades de Castilla-La Mancha, Consejería de Educación y Ciencia, in Spain. It is also supported by the ESFINGE project (TIN2006-15175-C05-05) Ministerio de Educación y Ciencia (Dirección General de Investigación) / Fondos Europeos de Desarrollo Regional (FEDER) in Spain.

References

[1] A. Abdul-Rahman and S. Hailes, "Supporting Trust in Virtual Communities". In Proceedings of the 33rd Hawaii International Conference on Systems Sciences (HICSS), IEEE Computer Society, Vol. 6: pp. 1769-1777, 2000.

[2] Amazon , URL:http://www.amazon.com, 1996.

[3] K. Barber and J. Kim, "Belief Revision Process Based on Trust: Simulation Experiments". In 4th Workshop on Deception, Fraud and Trust in Agent Societies, pp. 1-12, Montreal Canada, 2004.

[4] A. Caballero, J. Botia, and A. Skarmeta, "A New Model for Trust and Reputation Management with an Ontology Based Approach for Similarity Between Tasks". In: Fischer, K., Timm, I. J., André, E., Zhong, N. (eds.), MATES LNCS 4196, pp. 172-183, 2006.

[5] J. Carbó, M. Molina, and J. Davila, "Trust Management through Fuzzy Reputation", In International Journal of Cooperative Information Systems, Vol. 12, No. 1, pp. 135-155, 2003.

[6] T. Davenport and L. Prusak, "Working Knowledge: How Organizations Manage What They Know". Project Management Institute. Harvard Business School Press. Boston, Massachusetts, 1997.

[7] eBay, URL: http://www.ebay.com. 1995.

[8] R. Guizzardi, L. Aroyo, and G. Wagner, "Help&Learn: A Peer-to-Peer Architecture to Support Knowledge Management in Collaborative Learning Communities". In Proceedings of XIV Brazilian Symposium on Computers in Education. Vol. 12, No. 1, pp. 29-36, Rio de Janeiro, 2003

[9] T. Huynh, N. Jennings, and N. Shadbolt, "FIRE: an integrated trust and reputation model for open multi-agent systems". In Proceedings of 16th European Conference on Artificial Intelligence (ECAI), pp. 18-22, 2004

[10] J. Sabater and C. Sierra, "Social REGRET, a reputation model based on social relations". In Proceedings of the Fifth International Conference on Autonomous Agents, Vol. 3, No. 1, pp. 44-56, 2002.

[11] J.P. Soto, A. Vizcaíno, J. Portillo-Rodriguez, and M. Piattini. "A Three Level Multi-agent Architecture to Foster Knowledge Exchange". In 19th International Conference on Software Engineering and Knowledge

Engineering (SEKE), pp. 565-569, Boston (USA), 2007.

[12]E. Wenger, Communities of Practice: Learning Meaning, and Identity. Cambridge University Press, 1998

[13] E. Wenger, R. McDermott, and W. Snyder, Cultivating Communities of Practice. Harvard Business School Press, 2002.

[14] G. Zacharia, A. Moukas, and P. Maes, "Collaborative Reputation Mechanisms in Electronic Marketplaces". In 32nd Annual Hawaii International Conference on System Science (HICSS), 1999.